

Wissensnetzwerke im Grid (WisNetGrid)
**Entwicklung von Methoden und
Werkzeugen für Workflowmodellierung
und -Ausführung**

Bericht zu Arbeitspaket 3.3

March 2012

Martin Junghans, Sudhir Agarwal
Karlsruher Institut für Technologie (KIT)



GEFÖRDERT VOM



Bundesministerium
für Bildung
und Forschung

Inhaltsverzeichnis

1	Einführung	2
2	Workflow-Modellierung im Grid	4
2.1	Das Prozessmodellierungswerkzeug Oryx	4
2.1.1	Architektur	5
2.1.2	Prozessmodellierung	6
2.1.3	Erweiterbarkeit	8
2.2	WisNetGrid Modellierungswerkzeug für Dienste und Workflows	10
2.2.1	Stencil Set für die WisNetGrid Beschreibungssprache .	10
2.2.2	Unterstützung zur semantischen Modellierung	12
2.2.3	Installation des Modellierungswerkzeugs	16
3	Workflow-Ausführung im Grid	17
3.1	Ausführung einzelner Elemente der Modellierungssprache . . .	18
3.2	Installation der Ausführungskomponente	24
4	Zusammenfassung	25

Kapitel 1

Einführung

Zu Beginn des WisNetGrid Projekts wurde im AP3 zunächst eine Architektur als Grundlage für die Interaktion zwischen verschiedenen Komponenten der Wissensschicht sowie zwischen Systemkomponenten und externen Komponenten entwickelt (vgl. Abbildung 1.1). In diesem Arbeitspaket wurden Dienste konzipiert und entwickelt, die eine allgemeine Infrastruktur für die Verwaltung von Beschreibungen von Diensten im Grid bereitstellen. Diese beinhalten Dienste für die Verwaltung und semantische Suche von Dienstbeschreibungen [9]. Die Verwendung von geeigneten Formalismen für die Beschreibung von Diensten stellte eine technische Herausforderung dar [5], da er einerseits möglichst viele Dienste semantisch beschreiben können soll (Ausdrucksmächtigkeit) andererseits effiziente Schlussfolgerungstechniken (Komplexität) nicht verhindern soll. Das letztere benötigt man z.B. für Suche nach geeigneten Diensten aus der Vielzahl von Diensten im Grid in vertretbarer Zeit [9].

Hat man die geeigneten Dienste gefunden, komponiert man die Dienste zu einem Prozess, den man dann ausführen möchte. Es bedarf also Techniken und Werkzeugen für die Modellierung und Ausführung von dienst-orientierten Workflows. Nachdem die Anforderungen an solche Werkzeuge gesammelt wurden und Methoden zur dienst-orientieren Workflowkomposition entwickelt wurden [6], werden im vorliegenden Bericht zum Arbeitspaket 3 die Werkzeuge zur Dienstbeschreibung, Workflowkomposition und Ausführung vorgestellt.

In Kapitel 2 stellen wir das graphische Modellierungswerkzeug zur formalen Beschreibung von Diensten und Workflows vor. Ausgehend von einem

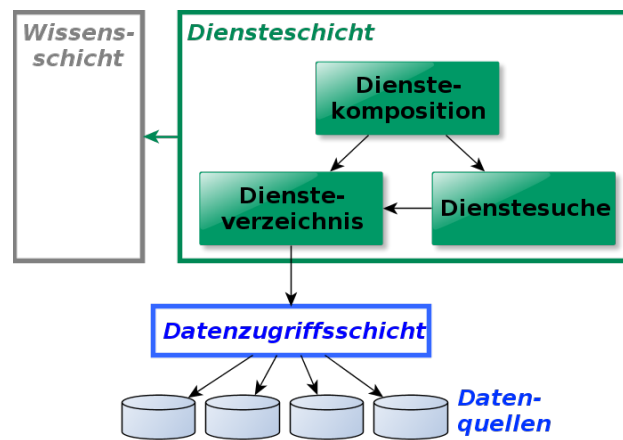


Abbildung 1.1: Komponenten der WisNetGrid Diensteschicht in Zusammenhang mit der Wissens- und Datenzugriffsschicht.

existierenden Werkzeug werden die im WisNetGrid Projekt entwickelten Erweiterungen beschrieben. Eine Anwendung zur Ausführung von Workflows wird in Kapitel 3 beschrieben. Abschließend fassen wir in Kapitel 4 diesen Bericht zusammen.

Kapitel 2

Workflow-Modellierung im Grid

Dieses Kapitel gibt einen Überblick über das im WisNetGrid Projekt entwickelte Werkzeug zur semantischen Dienst- und Workflowmodellierung. Es wurde ein existierendes Werkzeug zur Prozessmodellierung für die Anforderungen der semantischen Modellierung von Diensten und Workflows im Grid erweitert. Im Abschnitt 2.1 stellen wir zunächst das verwendete Prozessmodellierungswerkzeug vor und begründen die Auswahl dieses Werkzeugs als Grundlage unserer Entwicklungen. Die in WisNetGrid entstandenen Erweiterungen sind in Abschnitt 2.2 beschrieben.

Unser Werkzeug unterstützt die semantische Modellierung von Workflows als auch von potentiell komplexen Verhalten von Diensten gleichermaßen. Durch die Verwendung einer Beschreibungssprache für Dienste und Workflows können Dienste und Workflows bei der Modellierung komplexerer Workflows eingebettet und wiederverwendet werden. Diese Beschreibungssprache wurde bereits in [7] sowie im Bericht D3.2.2 [4] ausführlich erläutert.

2.1 Das Prozessmodellierungswerkzeug Oryx

Bei dem zu erweiternden Editor zur Prozessmodellierung handelt es sich um den Open Source Prozesseditor Oryx des Hasso Plattner Instituts [1]. Der Editor wurde von einer Studentengruppe im Jahr 2006/07 im Zuge ihrer Bachelorarbeit entwickelt und befindet sich seitdem in ständiger Weiterentwicklung. Informationen zu dem Projekt sind unter <http://oryx-project.org>

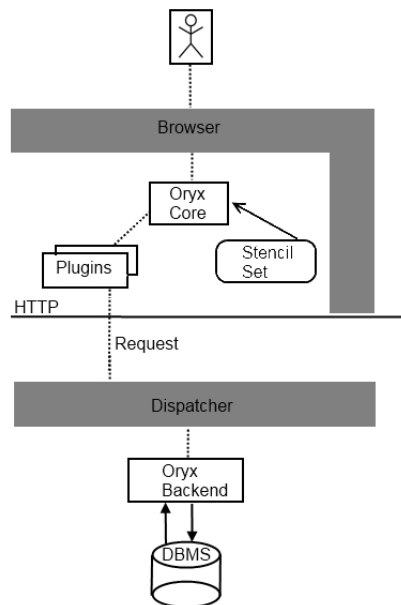


Abbildung 2.1: Architektur des Oryx Prozess-Editors.

zu finden. Oryx zeichnet sich durch eine sehr gute Erweiterbarkeit und Anpassungsfähigkeit aus und unterstützt bereits mehr als 25 Modellierungssprachen und Diagrammarten (z.B. BPMN, BPEL, Petri Netze, UML). Neue Prozesssprachen können mittels einer Beschreibungssyntax nachträglich integriert werden und der funktionale Umfang des Editors kann durch eigene *Plugins* erweitert werden. Diese leichte Erweiterbarkeit von Oryx zeichnet es als ideale Grundlage für die Weiterentwicklung zu einem Modellierungswerkzeug von Diensten und Workflows im Kontext von WisNetGrid.

2.1.1 Architektur

Der Editor besteht aus einer Browser-basierten graphischen Benutzerschnittstelle, die für die Modellierung und Darstellung der Prozessmodelle zuständig ist und einem server-seitigen *Backend*, welches die Prozessmodelle persistiert [11]. Eine Übersicht über die Architektur ist in Abbildung 2.1 zu finden.

Der Benutzer bedient den Editor über einen Web-Browser. Diese graphische Nutzerschnittstelle ist mit JavaScript programmiert und nutzt das JavaScript Framework ExtJS [2], eine Bibliothek für interaktive Webanwen-

dungen. Die Darstellung des Clients erfolgt mittels eines Browsers und besteht aus dem *Oryx Core*, welcher für das Layout der Anwendung zuständig ist. Die Funktionalitäten des Clients werden selbst auch über weitere Plugins zur Verfügung gestellt. Die modellierten Prozessmodelle werden im Client in der JavaScript Object Notation (JSON) beschrieben. Da der Client dieses Format interpretieren kann, ist es möglich Prozesse gleich aus dem Client zu exportieren auch zu importieren. Für die Persistierung wird die Prozessbeschreibung im JSON Format asynchron an das Backend geschickt, welches diese Informationen in einer Postgres Datenbank speichert. Ein gespeichertes Prozessmodell kann über eine eindeutige URL direkt identifiziert und zum Editieren geladen werden. Dazu wird der Editor geöffnet und die Daten werden vom Backend wieder in Form eines JSON Arrays an den Client geschickt und dort für die Darstellung des Prozessmodells verwendet. Stencil Sets dienen der Definition weiterer Modellierungssprachen unabhängig von der Implementierung des Editors. In Abschnitt 2.1.3 gehen wir genauer auf diese Möglichkeit ein.

2.1.2 Prozessmodellierung

Die Modellierung eines Prozesses erfolgt mittels Drag & Drop. Wie in Abbildung 2.2 fuer einen Prozessmodell in der Sprache BPMN beispielhaft dargestellt, werden verschiedene Arten von Aktivitäten aus dem *Shape Repository* (im linken Bereich der Anwendung) ausgewählt und dem Prozessmodell hinzugefügt. Die Kontrollstruktur des Prozesses wird durch das Verbinden der Aktivitäten mit Sequenz-Pfeilen modelliert. Der Datenfluss zwischen verschiedenen Aktivitäten wird mit Datenflusspfeilen modelliert. Diese Pfeile sind ebenfalls im Shape Repository auswählbar. Eigenschaften des Prozesses oder der einzelnen Aktivitäten lassen sich in dem tabellarisch angeordneten Dialog am rechten Bildschirmrand der Anwendung eingeben.

Zur Vereinfachung der Prozessmodellierung beinhaltet Oryx auch Zusatzfunktionen wie Rückgängig machen, Wiederherstellen, sowie das Kopieren und Einfügen von Prozesselementen. Neben dem Export der Prozessmodelle in das JSON Format ist auch ein Export in RDF möglich. Die Visualisierung der Prozessmodelle kann im PDF- oder PNG-Format exportiert werden.

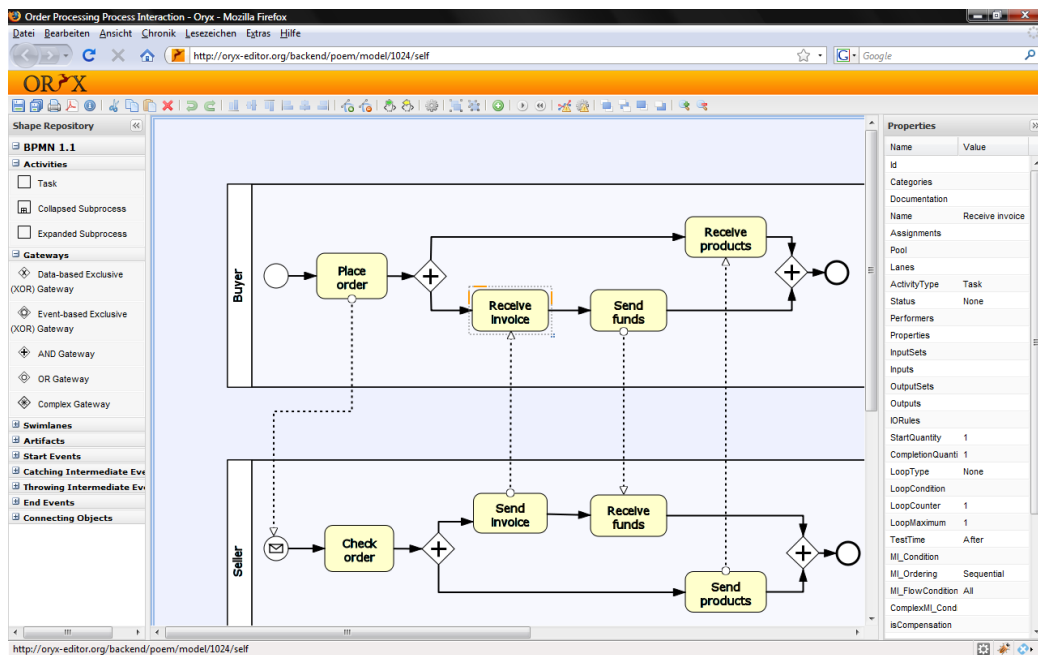


Abbildung 2.2: Modellierung eines Prozesses in BPMN 1.1 mit Oryx. (Quelle: <http://oryx-project.org>)

2.1.3 Erweiterbarkeit

Eine einfache Erweiterbarkeit des Oryx Editors wird vor allem durch zwei Features gewährleistet. Neue Prozesssprachen können anhand eines sogenannten *Stencil Sets* nachträglich in den Editor geladen werden. Daneben kann der funktionale Umfang des Editors durch selbst entwickelte server- als auch client-seitige *Plugins* erweitert werden. Da bereits vorhandene Plugins auch deaktiviert werden können, kann der gesamte Funktionsumfang des Oryx auf die eigenen Bedürfnisse angepasst werden.

Stencil Sets

Die Beschreibung einer neuen Prozesssprache anhand eines Stencil Sets erfolgt ebenfalls in Form einer JSON Datei und umfasst die Beschreibung der einzelnen *Stencils*, den Modellierungselemente einer Prozesssprache, sowie einem Regelwerk zur Einschränkung der Verwendung. Da die Beschreibung der Stencil Sets sehr umfangreich ist, wird in dieser Arbeit nur auf die Grundlagen eingegangen. Weitere technische Details können in der Bachelorarbeit von Nicolas Peters nachgeschlagen werden [13].

Beschreibung der Modellierungselemente einer Prozesssprache Bei der Beschreibung von einzelnen Elementen wird zwischen zwei Typen von Stencils unterschieden.

- **Nodes** stellen die Knoten in der grafischen Repräsentation dar. Bei einer Prozesssprache wie BPMN sind dies zum Beispiel die Aktivitäten und Gateways
- **Edges** verbinden die Nodes miteinander. Bei BPMN wäre dies zum Beispiel der Sequenzfluss oder der Datenfluss.

Für jedes Element muss eine Reihe von Eigenschaften definiert werden um es darstellen zu können. Dies umfasst zum Beispiel den anzuzeigenden Namen oder eine ID. Zudem enthält das Stencil Set Verweise auf Dateien, welche für die Darstellung der Elemente im Editor und im Menü gebraucht werden. Für die Darstellung im Editor wird eine Scalable Vector Graphics (SVG) Datei referenziert. SVG ist eine XML basierte Spezifikation zur Beschreibung von

zweidimensionalen Vektorgrafiken, welche die Skalierbarkeit der Elemente gewährleistet. Für die Darstellung der Elemente im Menü des Editors wird ein Verweis auf ein passendes Icon benötigt.

Bei der Modellierung können die Eigenschaften der Prozesselemente auch über ein Menü im Editor modifiziert werden. Für jedes Stencil können dazu eigene *Properties* im Stencil Set definiert werden. Diese Properties können neben einer normalen Typ-Definition wie String, Integer oder Float auch spezielle Eigenschaften haben. Zum Beispiel können in den Properties auch Boxen für die Auswahl des Datums definiert werden. Es gibt auch eine Möglichkeit ein Property als vom Typ DiagramLink zu definieren. Der Wert dieser Property wird dann als URL gehandhabt und kann mit einem Verweis auf ein Element aus den SVG Grafiken im Editor verknüpft werden. Somit es möglich aus der Editorfläche des Editors heraus URLs zu öffnen. Dies ist vor allem für das Referenzieren auf Subprozesse relevant.

Verbindungsregeln Zwischen Elementen Innerhalb eines Stencil Sets müssen Regeln definiert werden, welche die Verbindung zwischen den Nodes steuern. Oryx ist so konzipiert, dass alle nicht definierten Verbindungen verboten sind. Um eine Verbindung zwischen Elementen zu ermöglichen, muss eine explizite Regel erstellt werden. Stencils können auch gruppiert werden, um die Definition von Regeln zu erleichtern. Somit kann eine syntaktische Korrektheit der modellierten Prozessmodelle sicher gestellt werden.

Plugins

Neben der Erweiterung des Oryx Editors durch neue Prozesssprachen ist es auch möglich weitere Programmfunktionalitäten für den Editor hinzuzufügen. Diese können in Form von Plugins in den Editor eingebunden werden. Hierbei wird zwischen server-seitigen und client-seitigen Plugins unterschieden. Bei den server-seitigen Plugins handelt es sich um Java Servlets. Plugins für den Client werden in JavaScript geschrieben und müssen bei der Anwendung registriert werden. Geladene Plugins bieten ihre Funktionalität über eine standardisierte Schnittstelle an und haben dadurch ebenso Zugriff auf die Funktionalitäten von anderen Plugins.

2.2 WisNetGrid Modellierungswerkzeug für Dienste und Workflows

Dieser Abschnitt befasst sich mit der konzeptionellen Realisierung, um Dienste und Workflows im Kontext von WisNetGrid mit Oryx modellieren zu können und mit semantischen Erweiterungen annotieren zu können. Aufbauend auf der semantischen Beschreibungssprache [4] werden die nötigen Elemente und Anforderungen für die Entwicklung eines Stencil Sets identifiziert und beschrieben. Danach werden semantische Annotationen, aufbauend auf den Konzepten des Semantic Web, eingeführt und für die semantischen Erweiterung von Dienst- und Workflowbeschreibungen angewendet. Hierbei wird der Begriff der Ontologie eingeführt und erläutert. Danach werden die notwendigen Erweiterungskomponenten beschrieben.

2.2.1 Stencil Set für die WisNetGrid Beschreibungssprache

Um einen Workflow bzw. analog das Verhalten eines Dienstes im Oryx Editor modellieren zu können, muss die WisNetGrid Beschreibungssprache anhand eines Stencil Sets beschrieben und in den Editor integriert werden. Die Syntax und Sprachregeln führen zu einer Reihe von Anforderungen, welche das Stencil Set erfüllen muss. Alle Elemente, welche für eine Modellierung nötig sind, sind mit einer kurzen Funktionsbeschreibung als Bildschirmabschnitt der Anwendung in Abbildung 2.3 zusammengefasst.

Die Stencil Set Definition umfasst Aktivitäten vom Typ Eingabe (*Input Activity*), Ausgabe (*Output Activity*) und Berechnung (*Local Operation*). Zusätzlich wird das Startelement zum Notieren des eindeutigen Startpunkts eines Workflows bzw. einer Verhaltensbeschreibung angeboten. Das Element Ende notiert entweder das Ende eines gesamten Workflows oder eines einzelnen Ausführungspfades als Teil eines Workflows. Die Kontrollstruktur wird durch die eine bedingte Verzweigung (*Condition*), Nebenläufigkeit (*Parallel*) sowie einer Auswahl (*Choice*) modelliert. Existierende Dienste und Workflows können über das *Service*-Element eingebunden werden. Die bisher genannten graphischen Elemente werden durch Nodes im Stencil Set definiert. Die folgenden Elemente sind Edges, die die Nodes miteinander verbinden können.

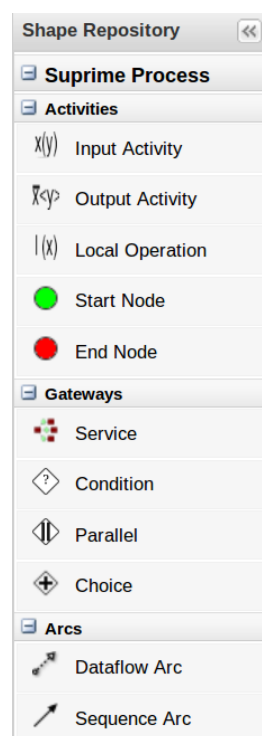


Abbildung 2.3: Palette der Modellierungselemente (Stencils) zur Beschreibung von Workflows und komplexen Dienstverhalten.

Sequenz- (*Sequence Arc*) und Datenfluss (*Dataflow Arc*) werden durch die Verbindung von Aktivitäten bzw. Gateways in Form von gerichteten Pfeilen dargestellt. In den Sequenzfluss müssen die Aktivitäten eingebunden werden. Das bedeutet, der Sequenzfluss verbindet diejenigen Elemente, welche nacheinander ausgeführt werden. Der Datenaustausch zwischen Aktivitäten findet über das Senden und Empfangen von Nachrichten über aktive Kanäle statt. Ein Kanal ist nur dann aktiv, wenn die Aktivitäten in denen sich Sender und Empfänger befinden, parallel ausgeführt werden und somit gleichzeitig sende- bzw. empfangsbereit sind. Dies soll in folgendem Beispiel noch einmal verdeutlicht werden.

2.2.2 Unterstützung zur semantischen Modellierung

In diesem Kapitel wird das Prinzip der semantischen Modellierung beschrieben. Darunter verstehen wir die Verhaltensmodellierung wie eben beschrieben in Verbindung mit der semantischen Beschreibung der sogenannten Prozessressourcen. Prozessressourcen sind Objekte (wie z.B. Parameter) die innerhalb der Aktivitäten vorkommen, verwendet werden und ggf. verändert werden.

Dienste und Workflows können analog zu Webseiten mit semantischen Annotationen beschrieben werden. Dazu wird das Wissen einer Domäne in Form einer Ontologie explizit und formal beschrieben. Die Bereitstellung des ontologischen Wissens wird über die Dienste und Werkzeuge der WisNet-Grid Wissensschicht realisiert. Die im Oryx Editor erstellten Beschreibungen lassen sich basierend auf dieser Funktionalität mit zusätzlichen formalen Informationen anreichern. Durch diese, für Maschinen verarbeitbaren Informationen, kann ein erhöhter Automatisierungsgrad bei der Verwendung der Dienst- und Workflowbeschreibungen erreicht werden. Dadurch wird z.B. die automatisierte Suche nach Diensten und Workflows in der WisNetGrid Dienstschicht ermöglicht.

Verwendung der WisNetGrid Wissensschicht Wir benötigen die Funktionalitäten des Ontologieverzeichnisses zur Speicherung und Abfrage von Ontologien im Grid [3]. Für jede Modellierung eines Dienstes oder eines Workflow wird mindestens eine Domänen-Ontologie innerhalb des Editors vom Ontologieverzeichnis benötigt. Dazu wurde ein server- als auch ein client-seitiges Plugin entwickelt, um im Backend Ontologien von der Dienstschnitt-

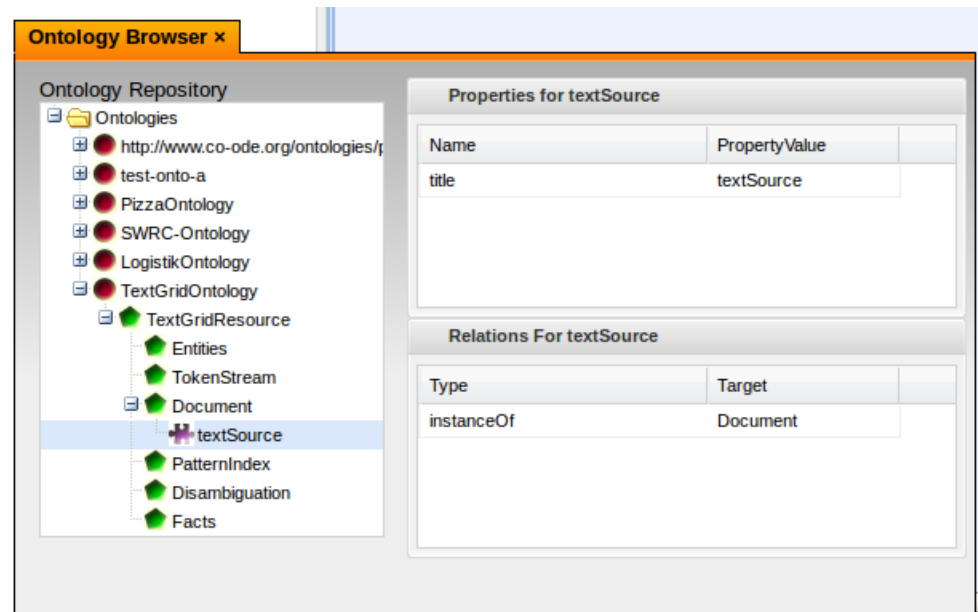


Abbildung 2.4: Anzeige von Ontologien zur semantischen Annotation von Prozessressourcen.

stelle des Ontologieverzeichnisses abzufragen (Details siehe [3]). Das Backend liest Ontologien ein und sendet eine interne Repräsentation der Ontologie an den im Web Browser laufenden Client. Die Kommunikation zwischen Client und Backend ist über Java Servlets realisiert. Das Plugin des Clients ist in JavaScript geschrieben und dient der Darstellung der Ontologien aus dem Verzeichnis innerhalb des Editors (vgl. Abbildung 2.4). Die Hauptbestandteile des Plugins sind ein Baum für die Darstellung der Klassen und Instanzen aus den Ontologien, sowie zwei Anzeigen für die Darstellung von weiteren Informationen über das aktuell selektierte Element im Baum. Somit werden auch Relationen zwischen Klassen oder Instanzen dargestellt.

Annotation semantischer Informationen Die Informationen aus den Ontologien müssen als Annotationen in der Dienst-/Workflow-Beschreibung gespeichert werden. Es soll sowohl möglich sein einen Dienst als Ganzes, aber auch einzelne Elemente des Verhaltens bzw. Workflows, annotieren zu können. Damit die Annotationen gespeichert werden können, müssen im Stencil Set hierfür Properties definiert werden, welche für die spätere Annotation bei der Modellierung vorgesehen sind.

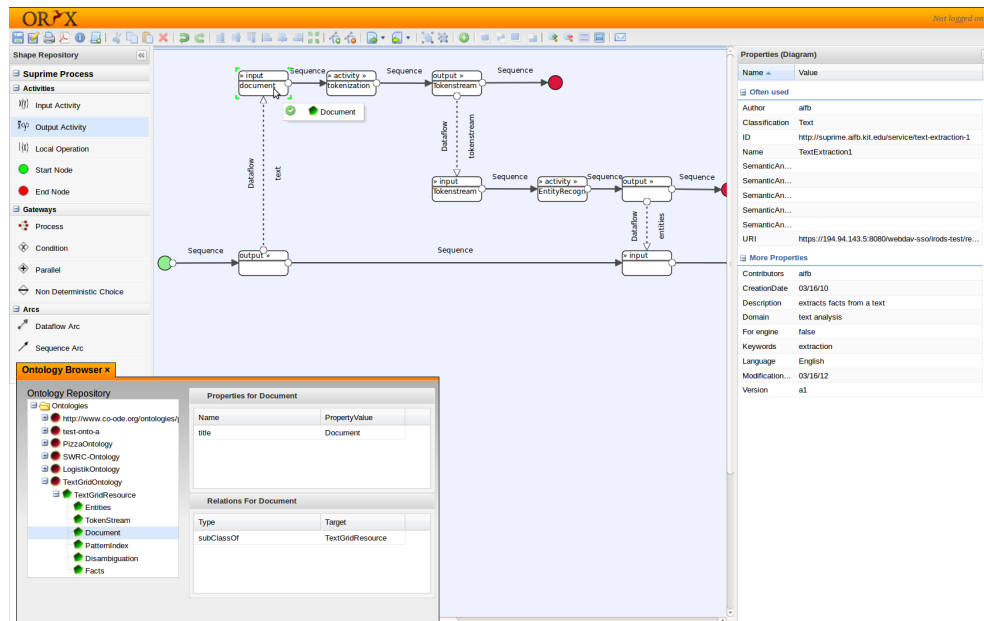


Abbildung 2.5: Semantische Modellierung eines wissenschaftlichen Workflows zur Wissensextraktion.

Wie in unteren linken Bereich der Abbildung 2.5 gezeigt, können die Konzepte der Ontologie basierend auf der Subklassenbeziehung nebst der Modellierungsfläche im *Ontology Browser* angezeigt werden. Mit Hilfe von Drag & Drop können Prozessressourcen annotiert werden. In dem Beispiel in Abbildung 2.5 wird das Konzept **Document** aus der Domänen-Ontologie ausgewählt und mit der Maus auf eine Eingabeaktivität gezogen. Daraufhin wird ein in einem Popup-Fenster die zu annotierende Prozessressource ausgewählt. In diesem Beispiel soll der Eingabeparameter **textSource** annotiert werden. Dadurch wird die Prozessressource **textSource** als Instanz der Klasse **Document** modelliert.

Komplexere semantische Beschreibungen von Prozessressourcen sowie Zusammenhänge zwischen verschiedenen Prozessressourcen werden durch Ontologie Axiome modelliert. Wie in Abbildung 2.6 gezeigt, wird im Wissensextraktionsbeispiel der Ausgabeparameter **tokenStream** über die Objektrelation **txtOnto:ExtractedFrom** in Beziehung mit den vorher eingegebenen Text **textSource** gesetzt. Daraus kann man automatisch schlussfolgern, dass der Tokenstrom aus dem Textdokument berechnet wurde.

Eigenschaften eines Dienstes werden im rechten Teil des Bildschirms spezifiziert. Falls keine Aktivität oder Verbindung innerhalb der Modellierungs-

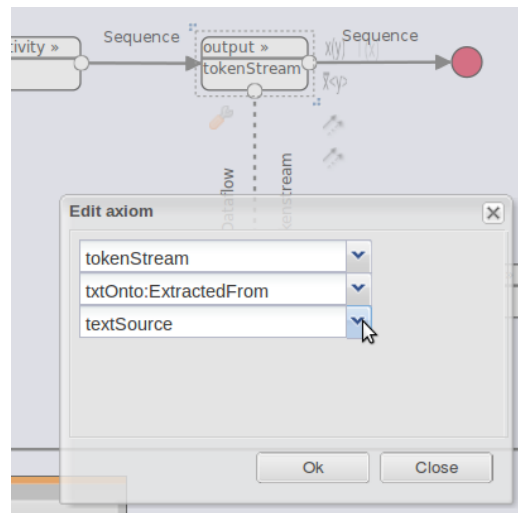


Abbildung 2.6: Semantische Modellierung von Zusammenhängen zwischen Prozessressourcen.

fläche selektiert wurde, können Metadaten als auch nicht-funktionale Eigenschaften des Dienstes eingegeben werden. Dazu wird eine Tabelle mit der ersten Spalte zur Benennung der Eigenschaft und der zweiten Spalte zur Eingabe eines Wertes angezeigt. Bei Metadaten sind die Werte der Eigenschaftsspalte vorgegeben. Nicht-funktionale Eigenschaften können vom Nutzer selbst aus einer Domänen-Ontologie gewählt werden.

Import und Export von Dienstbeschreibungen Das Modellierungswerkzeug verwendet das WisNetGrid Dienstverzeichnis [9] zum Speichern und Laden der semantischen Beschreibungen der Dienste und Workflows. Die im Editor erstellten Beschreibungen enthalten folgende Aspekte: Metadaten, nicht-funktionale Eigenschaften und die Beschreibung des Verhaltens durch das Prozessmodell. Die nicht-funktionalen Eigenschaften als auch das Prozessmodell verwenden Konzepte und Relationen aus einer oder mehreren Domänen-Ontologien. Da die Ontologien im Ontologieverzeichnis verwaltet werden, enthalten Dienst und Workflowbeschreibungen lediglich die IDs der verwendeten Ontologien. Mit Hilfe der IDs können die Ontologien aufgefunden werden.

Beschreibungen werden üblicherweise im Backend des Werkzeugs in einer relationalen Datenbank gespeichert. Für die Anbindung an das Dienstverzeichnis zur Speicherung der Beschreibungen im Grid wurde das Exportieren und Importieren von Beschreibungen durch client- und serverseitige Plugins

realisiert. Die client-seitigen Dialoge zum Im- und Export unterscheiden sich vom Laden-/Speichern-Dialogen, da zusätzlich ein Dienstverzeichnis gewählt werden muss und sich der Nutzer über Nutzernamen und Passwort authentifizieren muss. Im Backend wird die Beschreibung serialisiert bzw. de-serialisiert und mit Hilfe eines Web-Dienst-Clients an das Dienstverzeichnis kommuniziert.

2.2.3 Installation des Modellierungswerkzeugs

Unser Werkzeug ist eine Webanwendung, die als Web Archiv verteilt werden kann. Damit kann das Werkzeug über einen Servlet Container, wie z.B. Apache Tomcat, verfügbar gemacht werden. Im Backend ist im Betrieb die Anbindung an eine Postgres Datenbank notwendig. Die Einrichtung der Datenbank kann über ein Skript automatisch erfolgen. Die Anbindung an die Datenbank wird über eine Konfigurationsdatei des Modellierungswerkzeugs eingestellt. Nach dem Starten des Servlet Containers ist die Anwendung unter `http://<host>:<port>/backend/poem/repository` verfügbar.

Kapitel 3

Workflow-Ausführung im Grid

Eine Hauptmotivation für das WisNetGrid Projekt ist die Unterstützung von einer kreativen und flexiblen Erstellung und Ausführung von wissenschaftlichen Workflows in verschiedenen Wissenschaftsgemeinschaften unter der Nutzung der Grid-Infrastruktur [10]. Die Ausführung von Workflows ist neben der automatischen Suche auch eine weitere mögliche Verwendung der semantischen Dienst- und Workflowbeschreibungen, die durch die WisNetGrid Dienstschnittschicht betrachtet wird. Wir beziehen uns in diesem Kapitel auf die Ausführung von Workflows wie es bereits in Bericht D3.3.1 [6] motiviert und erarbeitet wurde. Im Folgenden beschreiben wir die Implementierung der WisNetGrid Workflow Ausführungskomponente.

Die Ausführung von Workflows wird direkt über einen Web Browser gesteuert. Die Schnittstelle zur Interaktion mit Nutzern wurde mittels HTML, JavaScript und Java Servlets erstellt und funktioniert in gängigen Web Browsern ohne zusätzliche Installation von weiterer Software. Die Web-Anwendung besteht zusätzlich aus einem Backend auf einem Server. Wir gehen davon aus, dass eine existierende Beschreibung eines Workflows ausgewählt wurde und diese Beschreibung im Backend der Ausführungskomponente verfügbar ist.

Die im Editor graphisch und deklarativ beschriebenen Workflows sind ausführbar unter der Annahme dass die Beschreibungen vollständig und korrekt sind. Die zur Ausführung notwendigen Informationen sind in der Beschreibung des Workflows enthalten. Auf Basis der Kontrollstruktur einer Beschreibung werden die einzelnen Aktivitäten sequentiell und synchron aufgerufen. Jedes Element einer Beschreibung gemäß der spezifizierten Reihenfolge abgearbeitet. Wird der Kontrollfluss durch Gateways (Parallel, Choi-

ce, Condition) beschrieben, wird die Ausführung ggf. in parallel laufenden Threads im Backend weitergeführt.

In den folgenden Abschnitten beschreiben wir die Elemente der Beschreibungssprache, die die Ausführungskomponente abgearbeitet werden. Die Ausführung eines Workflows beginnt beim Startelement.

3.1 Ausführung einzelner Elemente der Modellierungssprache

Eingabeaktivität

Eine Aktivität vom Typ einer Eingabe ist gekennzeichnet durch eine Menge von Eingabeparametern und einen Kommunikationskanal, über den die Parameter übertragen werden. Handelt es sich um eine Kommunikation zwischen zwei parallel laufenden Aktivitäten, werden die Ausgabeparameter der Ausgabeaktivität von der Ausführungskomponente im Backend entgegen genommen und temporär zwischengespeichert. Wenn möglich, werden die Ausgabeparameter den erforderlichen Eingabeparametern der wartenden Eingabeaktivität zugeordnet. Dies geschieht auf Basis der semantischen Informationen zur Typisierung der Parameter. Sollte eine automatische Vermittlung nicht möglich sein, wird der Benutzer während der Ausführung zur Eingabe der Werte der Eingabeparameter aufgefordert. Jeder Eingabeparameter ist durch dessen Bezeichner und den Typ gekennzeichnet. Der Benutzer kann den Wert eines Parameters in ein Textfeld eingeben. Währenddessen werden die Werte der Ausgabeparameter der Ausgabeaktivität in einem weiteren Browser-Tab angezeigt.

Findet die Kommunikation der Eingabeaktivität zwischen dieser Aktivität und dem Nutzer statt, dann wird die Browser-basierte Eingabemaske unmittelbar generiert von der Beschreibung der Eingabeaktivität generiert. Nach Eingabe der Parameterwerte setzt der Nutzer in beiden Fällen die Ausführung fort, indem der 'Continue' Knopf gedrückt wird. Die Werte der Eingabeparameter bleiben auch weiterhin für eine spätere Weiterverwendung in der Wissensbasis der aktuellen Instanz der Ausführungskomponente gespeichert.

Provide [Input Parameters](#)

Parameter	Type	Value
textSource	txtOnto:Document	<input type="text" value="https://roxxi.mmci.uni-saarland.de/data/einstein.txt"/>
lan	txtOnto:Language	<input type="text" value="de_DE"/>

Abbildung 3.1: Web-Formular zur Eingabe der Werte der Eingabeparameter.

Abbildung 3.1 zeigt den generierten Dialog zur Eingabe der Werte der Eingabeparameter einer Eingabeaktivität. Die Parameternamen sowie der Parametertyp (Konzept einer Domänen-Ontologie) werden angezeigt. Der Wert des Parameters kann von Nutzer editiert werden.

Ausgabeaktivität

Bei einer Ausgabeaktivität können analog zur Eingabeaktivität zwei Fälle auftreten. Bei einer Kommunikation zwischen zwei Aktivitäten ist der Benutzer nicht involviert, falls die Parameter automatisch vermittelt werden können. Bei einer Kommunikation zwischen der Ausgabeaktivität und dem Benutzer wird lediglich eine Webseite mit der Liste der Parameter, deren Typen und deren Werten generiert und zur Ansicht angezeigt. Danach kann der Nutzer die Ausführung durch Klicken auf 'Continue'.

Berechnungsaktivität

Diese Aktivitäten bezeichnen den Aufruf einer Methoden im lokalen Kontext (im Gegensatz zum Aufruf externer Dienste). Eine solche Berechnung manipuliert die Ressourcen der aktuellen Workflows. Z.B. können eingegebene Parameter umformatiert werden oder auch auch Informationen aus zuvor durch externen Dienste berechnete Zwischenergebnisse extrahiert werden.

Da die Beschreibung des Workflows selbst die Implementierungsdetails der Methode nicht enthält, muss die Ausführungskomponente basierend auf dem eindeutigen Methodennamen im lokalen Kontext die Implementierung dieser Methode nachschlagen. Es könnten zum Beispiel die Methoden der Java Bibliothek `java.util` als Menge der verfügbaren Berechnungsaktivitäten angeboten werden.

Ende

Im Kontrast zum eindeutigen Startelement, welches zum Starten der Workflow-Ausführung verwendet wird, kann es mehrere Endelemente in einem Workflow geben. Jeder parallel laufende Thread des Workflows kann mit einem Endelement terminiert werden. Die Ausführungskomponente erzeugt beim Auftreten des Endes eine Meldung im aktuellen Tab des Browsers, das dieser Pfad beendet wurde und der dazugehörige Browser-Tab geschlossen werden kann. Der entsprechende Thread im Backend wird terminiert.

Dienstaufruf

Der Aufruf eines externen Dienstes beginnt mit der Bereitstellung der Eingabeparameter, die mit dem Aufruf übergeben werden. Danach folgt die externe Bearbeitung der Anfrage. Dies geschieht selbständig und unabhängig von der Ausführungskomponente. Anschließend werden die Rückgabewerte des Dienstaufrufs empfangen.

Die Werte der Eingabeparameter werden analog zur Abarbeitung einer Eingabeaktivität über eine dynamisch generierte Webseite bereitgestellt. Die Rückgabewerte werden analog zu einer Ausgabeaktivität in einer generierten Webseite angezeigt und für eine spätere Weiterverwendung in der Ausführungskomponente gespeichert. Aufwändiger gestaltet sich der Aufruf des externen Dienstes. Es gibt viele Möglichkeiten externe Dienste bzw. Web Dienste in Abhängigkeit von deren Art und technischen Implementierung aufzurufen. Dienste, die im Grid angeboten werden, können für verschiedene Middleware Plattformen entwickelt werden. Zum Beispiel können Rechenjobs auf der Basis von UNICORE-6 angeboten werden. Um von den Grid-spezifischen Implementierungsdetails der Rechenjobs abstrahieren zu können, geben wir vor, dass diese Jobs als Web Dienste angeboten werden. Basierend auf einer definierten Web Dienst-Schnittstelle können beliebige Jobs und Dienste im Grid in die Workflows eingebunden werden. Daher werden wir in den folgenden Abschnitten den Aufruf existierender Web Dienste betrachten.

Eine der derzeit einfachsten und daher populärsten Methode zur Bereitstellung von Web Diensten basiert auf dem REST Paradigma. REST ist das Akronym für Representational State Transfer [8]. Eine Internetadresse (URL) beschreibt genau einen Seiteninhalt als Ergebnis einer server-seitigen Aktion. Zum Aufruf eines Dienstes ist das HTTP Protokoll ausreichend. Allerdings

REST Web Service Invocation

Service: <https://roxxi.mmci.uni-saarland.de/app/martin/rest/>

Set Up REST-like Web Service Invocation

Specify the Method Path

ps://roxxi.mmci.uni-saarland.de/app/martin/rest/run/(runid)/status

Method: HTTP/GET Accept: application/json

Next

Provide Template Parameters

runid: 6

Provide Query Parameters

Query parameter 1 name value Add another query parameter

Query parameter 2 name value

Invoke

Abbildung 3.2: Web-Formular zur Eingabe der Werte der Parameter zum Aufruf eines REST-basierten Dienstes. Links wird die Dienst-URL spezifiziert. Werte von Template Parametern werden im mittleren Bereich eingegeben. Query Parameter können im rechten Bereich des Formulars hinzugefügt werden.

werden die Definitionen von HTTP und REST meist nicht strikt angewendet. Dies führt dazu, dass die Vorstellungen über REST auseinander gehen sowie dass sich vielfältige Arten der Implementierung eines REST-basierten Dienstes ergeben. SOAP-basierte Web Dienste sind ebenfalls weit verbreitet. Im Gegensatz zu REST, gibt es mehrere Protokolle, Definitionen und Rahmenwerke, die diese Art von Diensten charakterisieren.

Allerdings wurden nur wenige Rahmenwerke entwickelt, die einen Aufruf dieser Dienste ohne Vorwissen über die Dienstimplementierungsdetails erlauben. In der Ausführungskomponenten benutzen wir daher DAIOS (Dynamic and Asynchronous Invocation of Services, [12]) zum Aufruf von REST- und SOAP-basierten Web Diensten. Dieses Rahmenwerk unterstützt die dynamischen Aufruf ohne jegliche statische Komponenten wie zum Beispiel Service Stubs, Endpoint Interfaces oder Nachrichtenobjekte. DAIOS baut auf dem Apache Axis 2 Rahmenwerk¹ auf und erweitert dieses um die Fähigkeit dynamisch die Nachrichtenobjekte zu erzeugen. Zum Beispiel werden beim Aufruf SOAP-basierter Dienste die SOAP-Nachrichten zusammengebaut. Der Aufruf selbst wird durch die Wiederverwendung des Axis 2 Service Stacks realisiert.

Durch die Verwendung von Axis 2 im Backend von DAIOS sprechen wir die Empfehlung zur Verwendung des Axis 2 Rahmenwerks für die Erstellung von Web-Dienst-Schnittstellen für die Grid-basierten Dienste oder Jobs aus. Dadurch kann trotz verschiedener Interpretationen und Implementierungen der Web-Dienst-Spezifikationen davon ausgegangen werden, dass der Auf-

¹Apache Axis 2 – <http://ws.apache.org/axis2>

REST Web Service Invocation

Service: <https://roxxi.mmci.uni-saarland.de/app/martin/rest/>

Set Up REST-like Web Service Invocation

Specify the Method Path

Method: HTTP/POST Accept: application/json

Next

Invoke!

Provide Query param

Abbildung 3.3: Web-Formular zur Eingabe der Werte der Parameter zum Aufruf eines REST-basierten Dienstes. Neben der Dienst-URL können die für den Aufruf zu verwendende HTTP Methoden sowie das Format der erwarteten Antwort des Dienstes verändert werden.

ruf solcher Web Dienste ohne weitere Anpassungen von DAIOS möglich ist. Im Laufe der Entwicklung der Ausführungskomponente wurden auch weitere Bibliotheken, wie zum Beispiel die freie Jersey Bibliothek zur Erzeugung REST-basierter Web Dienst, erfolgreich getestet. Allerdings können wir an dieser Stelle nicht für eine vollständige Unterstützung aller möglichen Bibliotheken durch DAIOS.

Abbildung 3.2 zeigt die Eingabe der der Parameter für den Aufruf eines REST-basierten Dienstes. Im gezeigten Beispiel werden zwei mögliche Arten der Bereitstellung der Parameter berücksichtigt. Template Parameter sind als Platzhalter in der URL des Dienstaufrufs enthalten. Zum Beispiel, die gezeigt Dienst-URL enthält den Parameter `{runId}`. Im Web-Formular zur Bereitstellung der Parameterwerte wird daraufhin dynamische eine Eingabeaufforderung für den Wert vom Parameter `runId` generiert (gezeigt im mittleren Bereich des Bildschirmausschnitts in Abbildung 3.2).

Eine weitere Art von Parametern sind Query Parameter, welche an die Dienst-URL angehängt werden. Die Anzahl der möglichen Parameter ist nur begrenzt durch die Länge der daraus entstehenden URL. Query Parameter

werden in der Form `?param1=val1¶m2=val2` an die URL des Dienstes angehängt.

Beim Aufruf REST-basierter Dienste kann man zusätzlich noch die HTTP Methode auswählen. Z.B. wird wie in Abbildung 3.3 ein Dienst zur Wissensextraktion aufgerufen, so muss die Methode zum Erstellen eines Extraktionsjobs über die HTTP POST Methode aufgerufen werden. Der zu analysierende Text als Argument im Body der Nachricht bei einem HTTP POST Aufruf an den Dienst übermittelt.

Bedingungen

Eine bedingte Verzweigung besteht aus einer Menge von ausgehenden Pfaden. Für jeden dieser Pfade existiert eine Bedingung die entweder wahr oder falsch ist. Die Ausführungssemantik einer bedingten Verzweigung ist so definiert, dass genau ein Pfad aus der Menge der Pfade mit einer wahren Bedingung zur Fortsetzung der Ausführung ausgewählt wird. Die Ausführungskomponente evaluiert die Bedingungen der ausgehenden Pfade sequentiell aus. Zur Evaluierung wird das während der Ausführung gesammelte Wissen herangezogen. Eine Bedingung kann sich zum Beispiel auf die Werte der in der Ausführung zuvor aufgetauchten Ein- oder Ausgabeparameter beziehen. Trifft die Ausführungskomponente auf eine Bedingung die wahr ist, wird die Ausführung des zu dieser Bedingung gehörenden Pfades fortgesetzt. Andere Pfade werden nicht weiter betrachtet.

Parallelität

Die Ausführungskomponente kann auch mehrere Pfade zur gleichen Zeit abarbeiten. Nach dem Auftreten des Parallel Gateways werden Java Threads für jeden ausgehenden Pfad des Gateways gestartet. Jeder einzelne Pfad wird eigenständig abgearbeitet. Der durch den Datenfluss modellierte Nachrichtenaustausch zwischen Aktivitäten parallel laufender Pfade ist die einzige Abhängigkeit zwischen verschiedenen Pfaden. Ansonsten werden die Pfade asynchron ausgeführt.

Wahlmöglichkeiten

Das Element 'Choice' bezeichnet eine Verzweigung basierend auf einer Nutzeraktion. Ansonsten wird ähnlich wie bei der bedingten Verzweigung verfahren. Auf der aktuellen Webseite zur Ausführung werden dem Nutzer die einzelnen Wahlmöglichkeiten zur Wahl gestellt. Selektiert der Nutzer während der Ausführung eine Auswahl durch Klicken auf einem Link, so wird der zugehörige Pfad im Folgenden ausgeführt.

3.2 Installation der Ausführungskomponente

Die Ausführungskomponente wurde als Webapplikation entwickelt. In kompilierter Form kann die Anwendung in einem Java Servlet Container, wie zum Beispiel Apache Tomcat, zur Anwendung gebracht werden. Andere Abhängigkeiten bestehen weder server- noch client-seitig.

Kapitel 4

Zusammenfassung

In diesem Dokument haben wir einen Überblick über die in WisNetGrid entwickelten Werkzeuge zur semantischen Modellierung von Dienstverhalten und Workflows sowie zur Ausführung von wissenschaftlichen Workflows. Beide Werkzeuge zeichnen sich durch ihre schnelle Verwendbarkeit aus, da sie ohne Installation weiterer Anwendungen direkt im Web Browser gestartet und benutzt werden können.

Die Beschreibung des Verhaltens von komplexen Diensten und Workflows werden mit Hilfe eines leicht zu erweiternden Prozessmodellierungswerkzeugs aus der Open-Source-Community erstellt. In Rahmen des Projektes wurde dieser Editor für die Anforderung aus dem WisNetGrid aber auch aus den Grid-Community-Projekten entsprechend erweitert. So wurde zunächst der Editor um die einfache WisNetGrid Dienstmodellierungssprache aus [5] erweitert. Mit Hilfe der Dienste und Werkzeuge aus dem Arbeitspaket 2 [3], deren Integration in den Editor sowie der Erweiterungen zum Anzeigen, Annotieren und Editieren ontologische Informationen wurde eine semantische Beschreibung der Dienste und Workflows ermöglicht. Schließlich wurde der Editor an das Dienstverzeichnis angebunden.

Die Ausführung wissenschaftlicher Workflows geschieht zentral und interagiert mit dem Benutzer über eine Browser-basierte Nutzerschnittstelle. Sie betrachtet Dienste im Grid die sowohl über eine REST- als auch über eine SOAP-Schnittstelle angeboten werden können. Dadurch kann von Grid-spezifischen Implementierungsdetails abstrahiert werden und es lässt sich eine

breitere Menge von Diensten und Rechenjobs in die ausführbaren Workflows einbetten.

Literaturverzeichnis

- [1] ORYX Process Editor. <http://oryx-editor.org>.
- [2] Sencha ExtJS Framework. <http://www.extjs.com>.
- [3] Sudhir Agarwal, Patrick Harms, Rene Jäkel, and Carolin Michels. Wissensnetzwerke im Grid: D2.2.2 Prototypische Implementierung des Schlussfolgerungs- und Vermittlungsdienstes. Technical report, Karlsruhe Institute of Technology (KIT), Juni 2010.
- [4] Sudhir Agarwal, Patrick Harms, Rene Jäkel, and Carolin Michels. Wissensnetzwerke im Grid: D3.2.2 Semantische Beschreibungssprache für Web-Dienste. Technical report, Karlsruhe Institute of Technology (KIT), Juni 2010.
- [5] Sudhir Agarwal, Patrick Harms, Carolin Michels, Silke Molch, and Eva Radermacher. Wissensnetzwerke im Grid: D 3.2.1 Anforderungen an die semantische Beschreibungssprache für Web-Dienste. Technical report, Karlsruhe Institute of Technology (KIT), Dezember 2009.
- [6] Sudhir Agarwal, Rene Jäkel, Martin Junghans, Steffen Metzger, and Bernd Schuller. Wissensnetzwerke im Grid: D3.3.1 Untersuchung und Vergleich von bestehenden Workflow Werkzeugen bzgl. der Anforderungen. Technical report, Karlsruhe Institute of Technology (KIT), Juni 2011.
- [7] Sudhir Agarwal, Martin Junghans, and René Jäkel. Semantic Modeling of Services and Workflows for German Grid Projects. In CEUR Workshop Proceedings, editor, *Grid Workflow Workshop 2011 (GWW'11)*, Köln, Germany, March 2011.
- [8] Roy Thomas Fielding. *REST: Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

- [9] Martin Junghans and Sudhir Agarwal. Wissensnetzwerke im Grid: D3.2.6 Evaluierung der Suchfunktion. Technical report, Karlsruhe Institute of Technology (KIT), Juni 2011.
- [10] Martin Junghans, Sudhir Agarwal, and René Jäkel. WisNetGrid Service Layer - Enabling Creative Research Workflows. In *European Grid Infrastructure Community Forum 2012*, Garching, Germany, March 2012.
- [11] Stefan Krumnow, Gero Decker, and Mathias Weske. Modellierung von EPKs im Web mit Oryx. In Peter Loos, Markus Nüttgens, Klaus Turowski, and Dirk Werth, editors, *MobIS Workshops*, volume 420 of *CEUR Workshop Proceedings*, pages 5–17. CEUR-WS.org, 2008.
- [12] Philipp Leitner, Florian Rosenberg, and Schahram Dustdar. Daios: Efficient dynamic web service invocation. *IEEE Internet Computing*, 13(3):72–80, 2009.
- [13] Nicolas Peters. Oryx Stencil Set Specification (Bachelor’s thesis). Available online at <http://oryx-editor.googlecode.com/files/OryxSSS.pdf>, June 2007.